



Data Project Presentation

Tugberk Erdogmus



Contents

1

Problem
Definition

2

Data
Extraction and
Manipulation

3

Optimization
Rules and
Python

4

Visualization
with PowerBI

5

Summary



1-Problem Definition

- **Borusan Otomotiv** is the official importer and distributor of BMW vehicles in Turkey. The company operates in the automotive sector, specializing in the sales and after-sales services of premium BMW cars and motorcycles.
- The objective of this project is to optimize the placement and capacity of Electric Vehicle (EV) charging stations throughout Istanbul, Turkey. With the rising adoption of electric vehicles, there is a critical need to develop an efficient charging infrastructure that meets the growing demand.



Borusan Otomotiv

2- Data Extraction and Manipulation

- **Station Data:** The potential EV charging station locations and their capacities were provided by the customer, Borusan Otomotiv BMW.
 - Street, District
 - Latitude
 - Longitude
 - Capacity
- **District Data:** Comprehensive information about Istanbul's districts was gathered from various reliable sources:
 - **Coordinates:** District coordinates were extracted from a JSON file available on GitHub, providing accurate geographical data. We put some effort in because it was a nested dictionary.
 - **Population Data:** The latest population figures for each district were obtained from the official data website of the Istanbul Municipality.
 - We joined these two tables on district names.



2.1 How to turn Population into Demand

```
import numpy as np

# Given parameters
vehicle_ownership_rate = 0.25      # 25%
ev_penetration_rate = 0.02         # 2%
market_share = 0.20                 # 20%
vehicles_per_station_capacity = 20   # One unit serves 20 cars

# The number of EVs in each district
merged_df['Number_of_EVs'] = merged_df['Population'] * vehicle_ownership_rate * ev_penetration_rate




# The number of EVs using our customer's stations
merged_df['EVs_using_customer_stations'] = merged_df['Number_of_EVs'] * market_share

# The demand (number of capacity units needed) in each district
merged_df['Demand'] = merged_df['EVs_using_customer_stations'] / vehicles_per_station_capacity

# Round up the demand to the nearest integer, ensuring at least one unit of demand
merged_df['Demand'] = merged_df['Demand'].apply(lambda x: max(int(np.ceil(x)), 1))
```


2.1 How to turn Population into Demand

Demand Calculation with statistics and assumptions

-  We began by recognizing that approximately 25% of Istanbul's population owns a vehicle, reflecting the proportion of residents likely to own any type of car in the city. Among these vehicles, 2% are electric vehicles (EVs), which represents the current EV adoption rate within Istanbul's vehicle fleet.
-  Understanding our customer's position in the market, we noted that Borusan Otomotiv BMW is projected to capture 20% of the EV charging market in Istanbul. This market share indicates the portion of EV owners expected to use our customer's charging stations over competitors'.
-  Additionally, we accounted for the operational capacity of the charging stations. Each unit of charging station capacity can serve 20 different EVs throughout the day, considering factors like charging time and station availability. This turnover rate ensures that the stations are utilized efficiently. (This rate is 12 in EU countries 🖐️🖐️)

3- Optimization Rules and Python

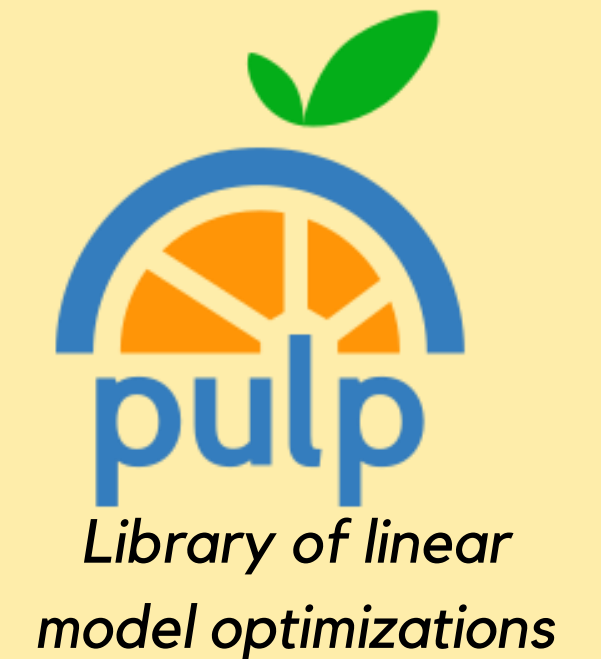
To efficiently meet the EV charging demand while minimizing costs, we developed an optimization model that determines the optimal placement and capacities of charging stations across Istanbul. The model incorporates several key components:

1- Objective Function

2- Decision Variables

3- Constraints

4- Parameters



3.1 Parameters

```
# Number of districts
num_districts = len(district_data)

# Number of stations
num_stations = len(stations)

# Number of selected stations
num_selected_stations = 50

# Distance threshold (theta) in kilometers
theta = 5

# Utilization rates
U_min = 0.1 # Minimum utilization rate
U_max = 1.0 # Maximum utilization rate

# Variable cost per slot
V = 100

# Station capacities
N_i = stations.Capacity.values

# Station fixed costs
station_fixed_costs = [population * 0.05 for population in district_data['Population']]

# Total station costs (fixed + variable)
station_total_costs = [station_fixed_costs[i] + V * N_i[i] for i in range(num_stations)]

# District demands
district_demands = district_data['Demand'].values.astype(int)

# District locations (latitude, longitude)
district_locations = district_data[['District_Latitude', 'District_Longitude']].values

# Station locations (latitude, longitude)
station_locations = list(zip(stations.latitude, stations.longitude))
```


3.2 Decision Variables

- **Station Selection Variables** (x_i): *Binary* variables indicating whether station i is selected (1) or not (0).
- **Capacity Allocation Variables** (y_{ij}): *Integer* variables representing the number of charging slots at station i allocated to district j .

```
# Decision variables
x = [pulp.LpVariable(f"x_{i}", cat='Binary') for i in range(num_stations)]
y = [[pulp.LpVariable(f"y_{i}_{j}", lowBound=0, cat='Integer')
      for j in range(len(district_data))]
      for i in range(num_stations)]
```

3.3 Objective Function

- **Minimize Total Cost:** The primary goal is to minimize the total cost, which includes both fixed costs (associated with establishing charging stations) and variable costs (related to the capacities of the stations).
- While doing that the number of functional stations had to be reduced to 30 from 50.

```
# Total cost per station
station_total_costs = [station_fixed_costs[i] + V * N_i[i] for i in range(num_stations)]
```

```
# Objective function
TotalCost = pulp.lpSum([station_total_costs[i] * x[i] for i in range(num_stations)])
prob += TotalCost
```

```
# Initialize the optimization problem
prob = pulp.LpProblem("EV_Charging_Stations_Optimization", pulp.LpMinimize)
```

3.4 Constraints

1- Demand Satisfaction: Ensure that the total charging capacity allocated to each district meets its estimated demand.

```
# 2. Demand satisfaction constraints
for j in range(len(district_data)):
    prob += pulp.lpSum([y[i][j] for i in range(num_stations)]) == district_demands[j], f"Demand_Satisfaction_{j}"
```

2- Capacity Limits: These constraints ensure that each station i , if selected, supplies at least the minimum utilization $U_{\min} * N_i$ and does not exceed the maximum utilization $U_{\max} * N_i$, maintaining a balanced use of capacity.

```
# Utilization rates
U_min = 0.1
U_max = 1.0
```

```
# 5. Utilization balance constraints
for i in range(num_stations):
    min_utilization = int(U_min * N_i[i])
    max_utilization = int(U_max * N_i[i])
    prob += pulp.lpSum([y[i][j] for j in range(len(district_data))]) >= min_utilization * x[i], f"Min_Utilization_{i}"
    prob += pulp.lpSum([y[i][j] for j in range(len(district_data))]) <= max_utilization * x[i], f"Max_Utilization_{i}"
```


3.4 Constraints

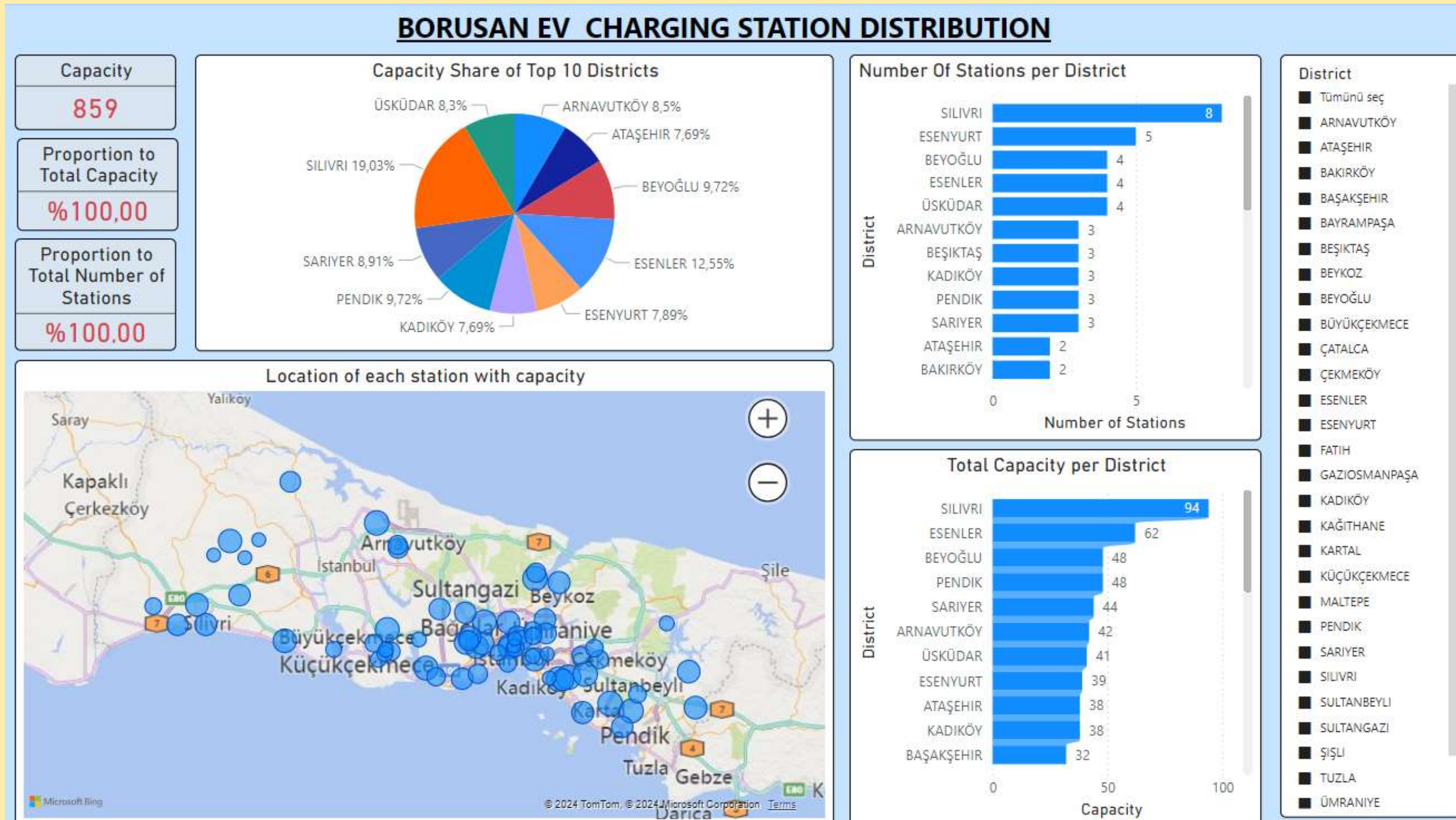
3- Service Distance Limitations: This constraint ensures that station i can only supply district j if the distance between them is less than or equal to θ ; otherwise, no supply is allocated from that station to the district.

```
# 4. Service distance constraint
for i in range(num_stations):
    for j in range(len(district_data)):
        distance = geodesic(station_locations[i], district_locations[j]).km
        if distance <= theta:
            s_ij = 1
        else:
            s_ij = 0
        # If s_ij is 0, y[i][j] must be 0
        prob += y[i][j] <= district_demands[j] * s_ij, f"Service_Distance_{i}_{j}"
```

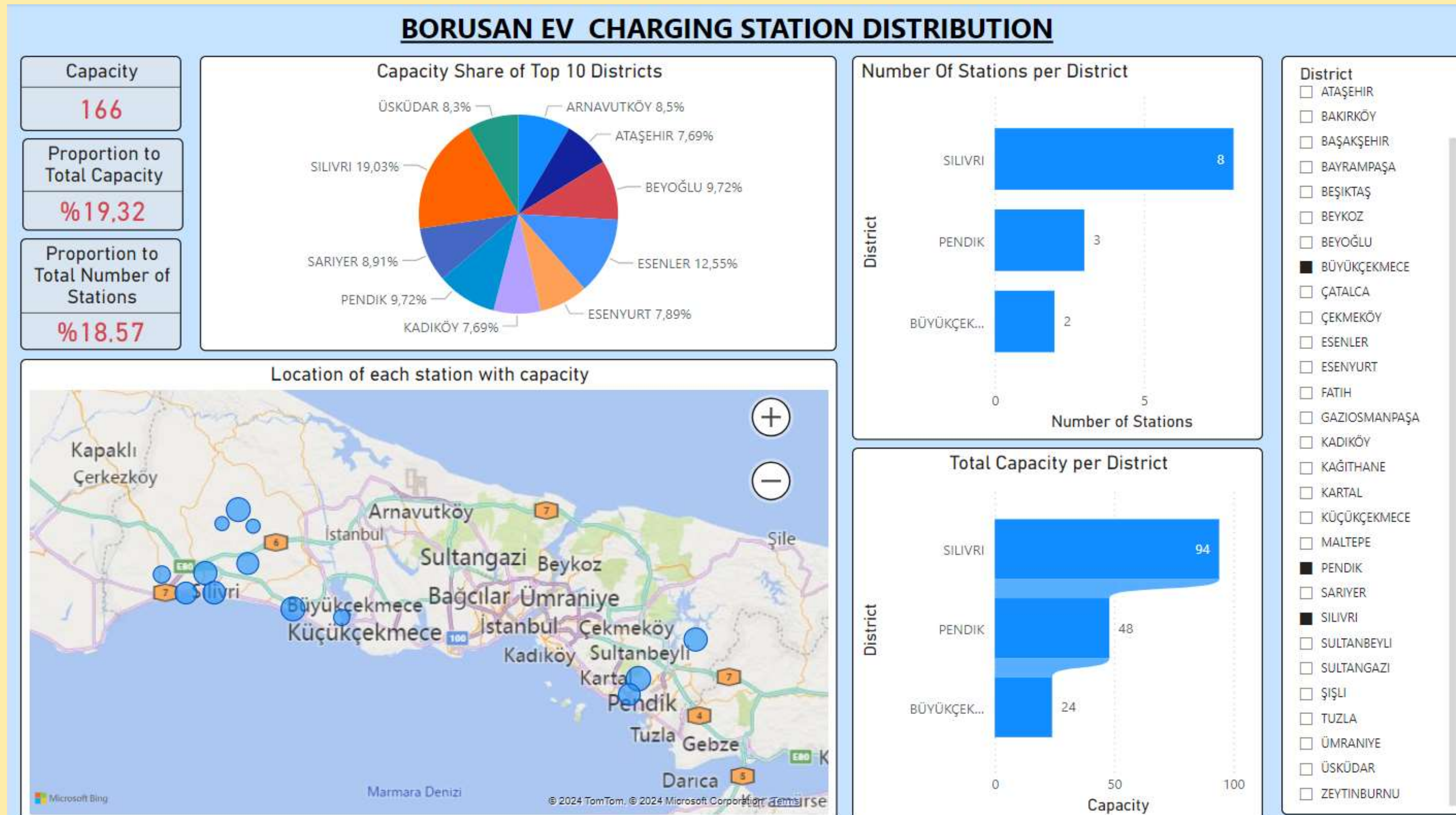
4- Station Selection Limit: This constraint ensures that the total supply from station i to all districts does not exceed the station's capacity $N_i[i]$ when the station is selected, indicated by the binary decision variable $x[i]$.

```
# 3. Supply constraints
for i in range(num_stations):
    prob += pulp.lpSum([y[i][j] for j in range(len(district_data))]) <= N_i[i] * x[i], f"Supply_Limit_{i}"
```

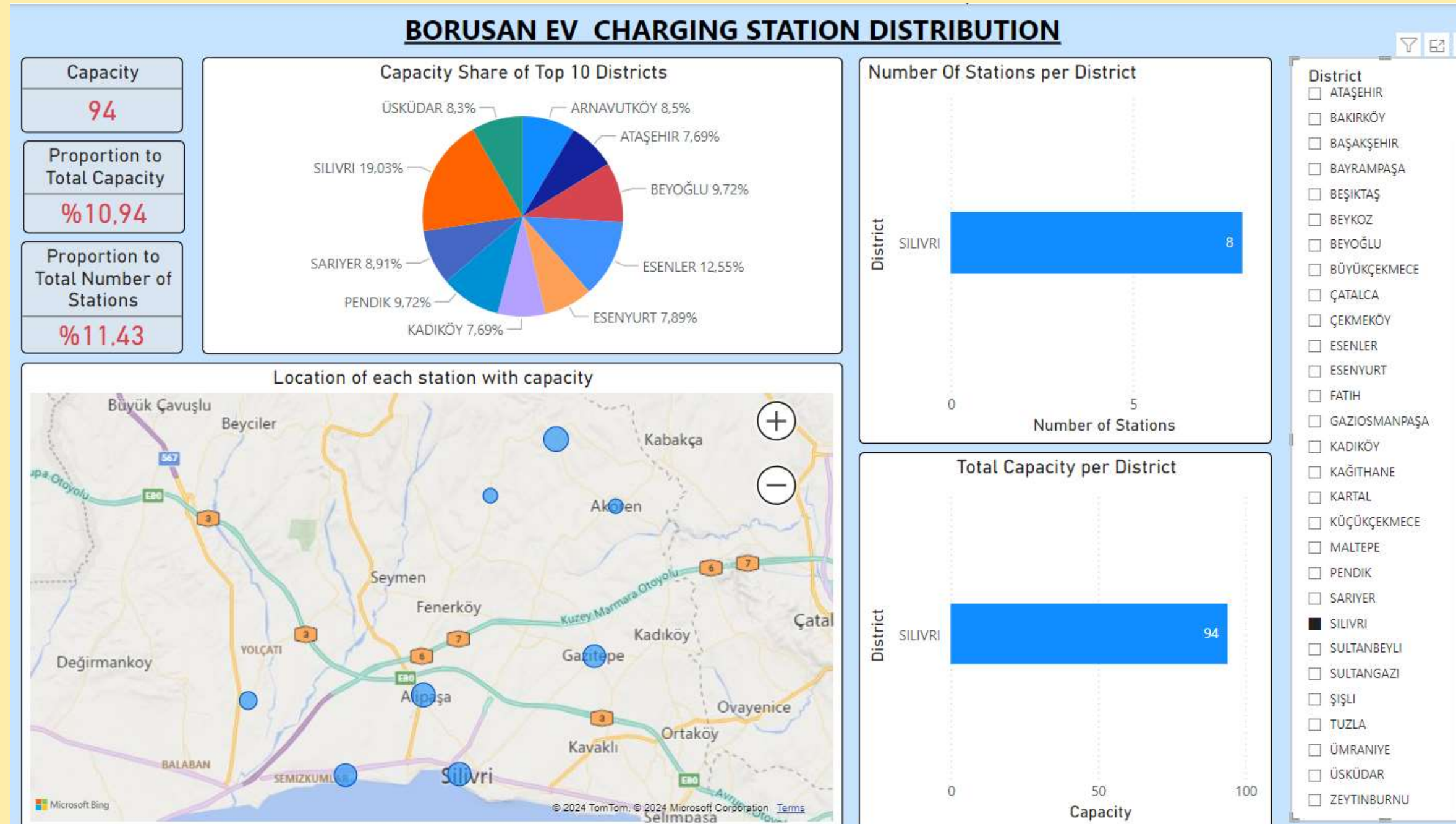

4. Visualization with PowerBI



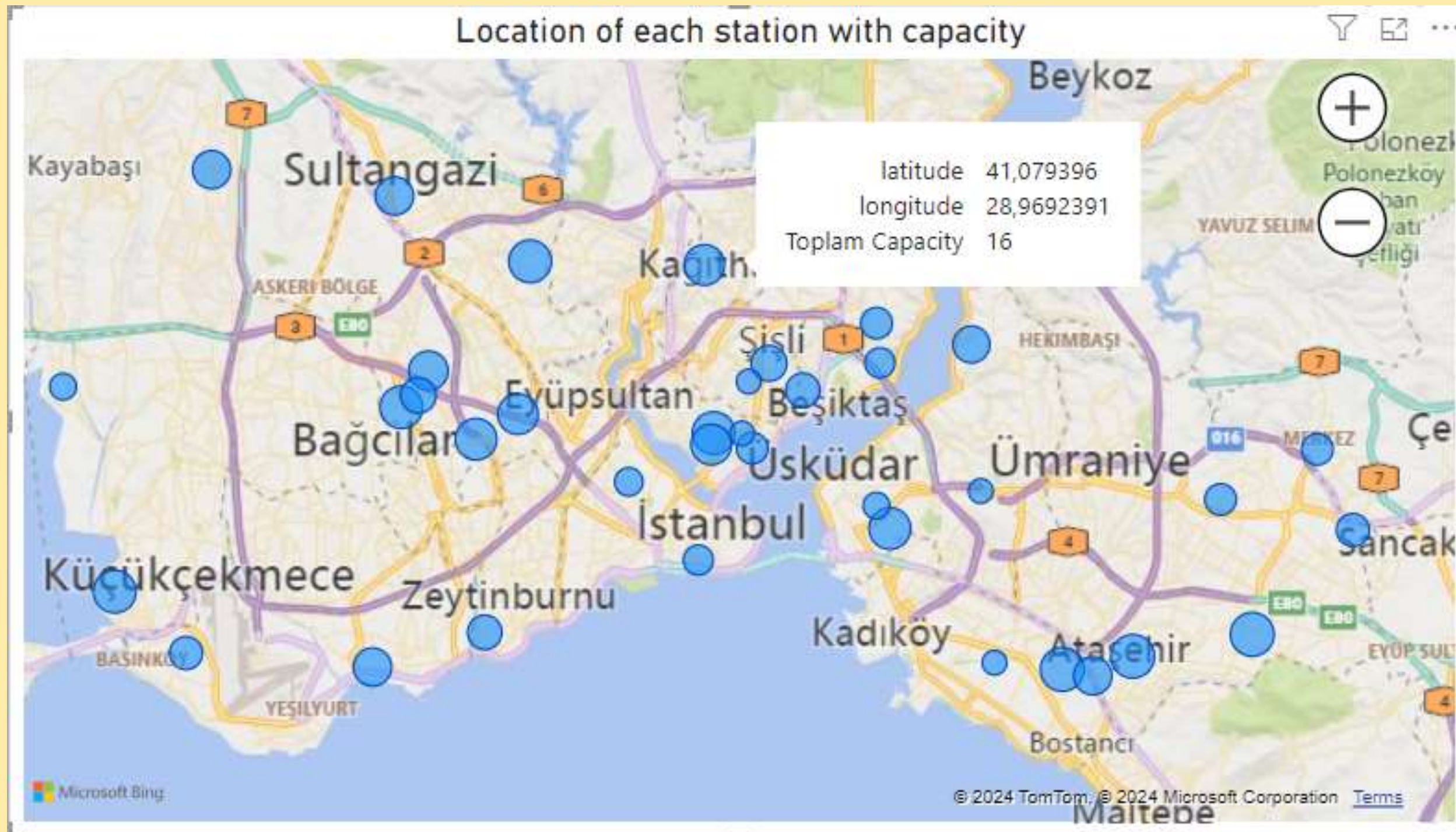
4. Visualization with PowerBI



4. Visualization with PowerBI



4. Visualization with PowerBI



4. Visualization with PowerBI

```
1 Percentage of Total Capacity =  
2 DIVIDE(  
3     SUM(Sheet1[Capacity]),  
4     CALCULATE(SUM(Sheet1[Capacity]), ALL(Sheet1[District])))
```

Veriler >>

Sheet1

☐ Σ Capacity

☐ ☐ Count of Stations

☐ District

☐ Σ latitude

☐ Σ longitude

☐ ☐ Percentage of Stations

☐ ☐ Percentage of Total Capacity

☐ Street

TOP10

☐ Σ Capacity

☐ District

☐ Σ latitude

☐ Σ longitude

☐ Street

5. Summary

- 1** Extracted data from various sources
- 2** Manipulated the data to ensure high quality input for the algorithm.
- 3** Utilized data to optimize the placement of EV charging stations while minimizing costs and ensuring all districts' demands are met.
- 4** Used PowerBI to present the results of our optimization in a way that the customer can easily understand and provide feedback on.

Questions

time